Induction of Classification Rules by Gini-Index Based Rule Generation

Han Liu[a,*], Mihaela Cocea[b]

[a]*School of Computer Science and Informatics, Cardiff University, Queen's Buildings, 5 The Parade, Cardiff, CF24 3AA, United Kingdom*
[b]*School of Computing, University of Portsmouth, Buckingham Building, Lion Terrace, Portsmouth, PO1 3HE, United Kingdom*

Abstract

Rule learning is one of the most popular areas in machine learning research, because the outcome of learning is to produce a set of rules, which not only provides accurate predictions but also shows a transparent process of mapping inputs to outputs. In general, rule learning approaches can be divided into two main types, namely, 'divide and conquer' and 'separate and conquer'. The former type of rule learning is also known as Top-Down Induction of Decision Trees, which means to learn a set of rules represented in the form of a decision tree. This approach results in the production of a large number of complex rules (usually due to the replicated sub-tree problem), which lowers the computational efficiency in both the training and testing stages, and leads to the overfitting of training data. Due to this problem, researchers have been gradually motivated to develop 'separate and conquer' rule learning approaches, also known as covering approaches, by learning a set of rules on a sequential basis. In particular, a rule is learned and the instances covered by this rule are deleted from the training set, such that the learning of the next rule is based on a smaller training set. In this paper, we propose a new algorithm, GIBRG, which employs Gini-Index to measure the quality of each rule being learned, in the context of 'separate

*Corresponding author
Email addresses: liuh48@cardiff.ac.uk (Han Liu), mihaela.cocea@port.ac.uk (Mihaela Cocea)
URL: hanliu94.wordpress.com (Han Liu), coceam.myweb.port.ac.uk (Mihaela Cocea)

and conquer' rule learning. Our experiments show that the proposed algorithm outperforms both decision tree learning algorithms (C4.5, CART) and 'separate and conquer' approaches (Prism). In addition, it also leads to a smaller number of rules and rule terms, thus being more computationally efficient and less prone to overfitting.

Keywords: Data Mining, Machine Learning, Decision Tree Learning, Rule Learning, Classification, If-Then Rules

---

1. Introduction

Due to the vast and rapid increase in the size of data, machine learning has been an increasingly important branch of artificial intelligence research. Rule learning is one of the most popular types of machine learning approaches. In general, rule learning can be achieved in two different approaches, namely, divide and conquer, and separate and conquer. The former type of rule learning is also known as Top-Down Induction of Decision Trees (TDIDT), which means that the nature of learning is to generate a set of rules represented in the form of a decision tree. Some popular algorithms include ID3 [32], C4.5 [34] and CART [3]. Also, the latter type of rule learning is also referred to as the covering approach, which means that the nature of the approach is to have a rule learned with a subset of training instances covered by this rule and then delete the subset of training instances prior to starting the learning of the next rule. Some popular algorithms include Prism [4] and Information Entropy Based Rule Generation (IEBRG) [20]. Due to the fact that rules learned by using the divide and conquer approach are represented in the form of a decision tree and the separate and conquer approach enables generating if-then rules directly from training instances [26], we refer the divide and conquer approach to as 'decision tree learning' and the separate and conquer approach to as rule learning (in a narrow sense) in the rest of this paper.

The main difference between decision tree learning and rule learning is that decision tree learning is aimed at attribute selection whereas rule learning is

aimed at the selection of attribute-value pairs. For example, an attribute x has two values 0 and 1, and $x = 0$ and $x = 1$ are two attribute-value pairs. In this context, attribute selection means to select the attribute x with its two values 0 and 1 as two different judgement criteria, whereas the selection of attribute-value pairs means to select either $x = 0$ or $x = 1$ as a judgement criteria. In decision tree learning, attribute selection leads to a non-leaf node being given and labeled the name of the selected attribute, and the node is also provided with several branches, each of which is attached a value of the attribute. In rule learning, the selection of an attribute-value pair leads to a rule term being appended to the left hand side of the rule. More details on attribute selection and the selection of attribute-value pairs are given in Section 2.

On the other hand, a decision tree learned through the divide and conquer strategy can be directly converted into a set of rules, each of which is extracted from a branch of the tree. However, rules learned directly from training instances through the separate and conquer strategy may not necessarily fit in a tree structure [4]. At each iteration of decision tree learning, attribute selection is done through measuring the importance of an attribute towards increasing the overall quality of a sub-tree being learned, i.e. increasing the overall quality of some rules extracted from some branches of a decision tree. In other words, the selected attribute would lead to the reduction of uncertainty for a tree to determine the class to which an instance belongs. However, the amount of uncertainty reduction is measured on average. For example, an attribute has a number of values and some values of the selected attribute may have a stronger ability to determine the class to which an instance belongs; other values of the attribute, however, may have a weaker ability in the determination of the class. The former case would lead to the reduction of uncertainty, while the latter case could lead to the increase of uncertainty. In fact, an attribute may have some of its values relevant to classifying instances, but the other values irrelevant [4], especially when an attribute is highly imbalanced with some values of very low frequency. In an extreme case, an attribute may have only one value relevant to classifying instances. On the basis of the above argumentation, decision tree

learning could result in some rules being of high quality but others being of low quality. In real applications, an instance is typically classified through using a single rule, and it is likely to give a wrong classification if the quality of this rule is low.

Due to the above problem of decision tree learning, we focus on rule learning methods in this paper through the separate and conquer strategy. The main motivation is that the selection of attribute-value pairs at each iteration of rule learning is done by measuring the importance of the attribute-value pairs towards increasing the quality of each single rule. In other words, the selected attribute-value pair would lead to the reduction of uncertainty for a single rule to determine the class to which an instance belongs. The contributions of this paper include the following:

- We propose a new algorithm, i.e. Gini-index based rule generation (GIBRG), which uses Gini-Index in rule learning for the selection of attribute-value pairs by measuring the amount of increase in terms of the quality of a single rule being learned.

- We compare the accuracy of the GIBRG algorithm with existing popular approaches. Experiments show that GIBRG leads to an improvement in classification accuracy compared with popular decision tree learning algorithms, such as C4.5 and CART, as well as rule-learning approaches such as Prism.

- We analysed in depth the difference between decision tree learning and rule learning in terms of the nature of the two types of learning approaches. In particular, we proved through experimental studies that the nature of rule learning can lead to the production of a fewer number of simpler rules compared with decision tree learning. Moreover, GIBRG results in a smaller number of simpler rules even when compared with Prism, a popular rule learning algorithm.

The rest of this paper is organized as follows: Section 2 provides the theoret-

ical preliminaries regarding the essence of the two types of learning approaches and the heuristics used for decision tree learning and rule learning. In Section 3, we review decision tree learning and rule learning, and analyse in depth why the nature of the separate and conquer approaches addresses some of the weaknesses of the divide and conquer approaches. In Section 4, we describe in detail the proposed GIBRG algorithm based on the Gini-index heuristic function in the context of rule learning. In Section 5, we report an experimental study to validate the Gini-index based rule generation (GIBRG) algorithm. Section 6 provides the summary of the contributions of this paper and further research directions.

2. Theoretical Preliminaries

This section presents theoretical preliminaries related to rule based systems in general, and to the difference between decision tree learning and rule learning in terms of the nature of their learning strategies, in particular. Several heuristic functions used in the learning process, such as entropy [37], information gain [15] and Gini-index [11], are described in detail.

2.1. Rule Based Systems

A rule based system is a special type of expert systems, which consists of a set of rules and can be learned through the divide and conquer strategy or the separate and conquer strategy. The following rule set, which consists of four rules, is provided below for illustrative purpose:

- Rule 1: if $x_1 = 0$ and $x_2 = 0$ then $class = 0$;
- Rule 2: if $x_1 = 0$ and $x_2 = 1$ then $class = 0$;
- Rule 3: if $x_1 = 1$ and $x_2 = 0$ then $class = 0$;
- Rule 4: if $x_1 = 1$ and $x_2 = 1$ then $class = 1$;

In this example, each rule has its antecedent and consequent. In particular, the left hand side (if part) of a rule is its antecedent, e.g. '$x_1 = 0$ and $x_2 = 0$', and the right hand side (then part) of a rule is its consequent, e.g. '$class = 0$'. Also, each antecedent is a conjunction of rule terms, e.g. '$x_1 = 0$' and '$x_2 = 0$' are two rule terms that make up a rule antecedent. Moreover, the rules are used to do classification so they are referred to as classification rules. In this context, the consequent of a rule is a reflection of the class assigned to any instances that are covered by this rule. For example, if an instance meets the two conditions that '$x_1 = 0$' and '$x_2 = 0$', then the instance is assigned the class '0'. In this case, we say that this rule (if $x_1 = 0$ and $x_2 = 0$ then $class = 0$;) is firing for the instance.

2.2. Procedure of Decision Tree Learning

The learning of a decision tree involves a repeated process of attribute selection. In particular, an attribute is selected to label the root node of a decision tree as a judgement criteria, and each branch starting from this node is attached a value or an interval of the selected attribute, which represents a judgement outcome. Each branch of a decision tree ends at an internal node leading to the growth of the tree by learning a subtree, or at leaf node labelled with a class leading to the termination of growing this branch. When a leaf node is given to end a branch, it typically indicates that the rule extracted from the tree branch has covered such a subset of training instances that belong to the same class and that the stopping criteria is satisfied. In practice, a leaf node may be given due to the case that the tree branch has reached its maximum length, i.e. the length of a branch must not exceed the number of attributes given for the data set. When a tree branch reaches its maximum length, all the attributes have been examined for partitioning a training set towards having each subset of training instances of the same class, but unfortunately the subset of training instances covered by a tree branch still ends up with belonging to different classes. In this case, the leaf node is labelled the majority class (the class to which the majority of the covered training instances belong) and the stopping criteria is

Algorithm 1: Decision tree learning [13]

Input : A set of training instances, attribute A_i, where i is the index of the attribute, value V_j , where j is the index of the value

Output: A decision tree T

```
1  if the stopping criterion is satisfied then
2  |  create a leaf that corresponds to all remaining training instances
3  end
4  else
5  |  choose the best (according to some heuristics) attribute A_i
6  |  label the current node with A_i
7  |  for each value v_j of the attribute A_i do
8  |  |  label an outgoing edge with value v_j
9  |  |  recursively build a subtree based on a subset of training instances
   |  |   that meet the condition 'A_i = v_j'
10 |  end
11 end
```

also treated as satisfied. The procedure of decision tree learning is described in Algorithm 1.

In line 5 of Algorithm 1, ID3 and C4.5 are designed to select the best attribute based on entropy as defined in Eq. 4, Section 2.4.1, or information gain as defined in Eq. 5, Section 2.4.2. In other words, the attribute that contributes to minimized entropy or maximized information gain would be selected at each iteration. Also, CART, which stands for classification and regression tree, is designed to select the best attribute based on Gini-index as defined in Eq. 9, Section 2.4.3. In other words, the attribute that contributes to minimized Gini-index would be selected at each iteration. More details on these popular decision tree learning algorithms can be found in [11, 25]; details of the heuristic functions mentioned above, i.e. entropy, information gain and Gini-index, are given in Section 2.4.

Algorithm 2: Rule learning [13]

Input : A set of training instances T

Output: An ordered set of rules RS

```
1 while T ≠ φ do
2 |   generate a single rule R from the training set T
3 |   delete all instances covered by rule R
4 |   if the generated rule R is not good then
5 |   |   generate the majority rule and empty the training set T
6 |   end
7 end
```

2.3. Procedure of Rule Learning

The learning of if-then rules involves a repeated process of attribute-value pairs. In particular, an attribute-value pair is selected to be appended to the left hand side of a rule, i.e. a rule is specialised by adding rule terms (attribute-value pairs) to its left hand side until the stopping criterion is satisfied. When the learning of a rule is stopped (completed), the rule would have covered a subset of training instances of the same class. Following the completion of learning a single rule, all instances covered by this rule need to be deleted from the training set prior to the start of learning the next rule. Learning of a rule set is completed once all training instances have been covered by at least one of the learned rules. The procedure of rule learning is described in Algorithm. 2.

In line 2 of Algorithm. 2, learning of a rule is done by heuristically selecting an attribute-value pair on an iterative basis towards specialising the rule. Line 4 indicates that the learning of a rule is stopped in the case that the rule has reached its maximum length, i.e. the length of a rule (the number of rule terms) must not exceed the number of attributes. When a rule being learned reaches its maximum length, all the attributes have been examined for specialising this rule but the training instances covered by this rule still belong to different classes. In this case, the majority class (the class to which the majority of the covered

training instances belong) is assigned as the consequent of this rule. Thus, this situation is treated in the same way as in decision tree learning.

Algorithm 3: Prism Algorithm [26]

Input : a training set T, a subset $T' \subseteq T$, an attribute set AS, an instance $t \in T$, dimensionality d, an attribute A_x, an attribute value v_{xm}, class C_i, number of classes n, max-probability p_{max}

Output: a rule set RS, a result set of instances T" covered by a rule $R \in RS$

```
1  Initialize: T' = T, T" = T', i = 0, p_max = 0;
2  for i < n do
3      while ∃t : t ∈ T' ∧ t ∈ C_i do
4          while ∃t : t ∈ T' ∧ t ∉ C_i do
5              x = 0; j = 0; p_max = 0; while x < d do
6                  for each value v_xm of A_x do
7                      Calculate P(C_i|A_x = v_xm);
8                      if P(C_i|A_x = v_xm) > p_max then
9                          p_max = P(C_i|A_x = v_xm); j = x; k = m;
10                     end
11                 end
12                 x++;
13             end
14             assign A_j = v_jk to R as a rule term; AS= AS - {A_j}; d= d−1;
15             ∀t : T" = T" − {t}, if t ∈ T' and t does not comprise A_j = v_jk;
16         end
17         RS= RS ∪ {R}; T'= T' - T";
18     end
19     T' = T; i++;
20 end
```

Regarding heuristic selection of attribute-value pairs, the Prism algorithm

is designed to select an attribute-value pair iteratively based on the conditional probability of a class given an attribute-value pair. In particular, the attribute-value pair, which contributes to maximizing the confidence of the rule being learned, would be selected as a rule term to be appended to the left hand side of this rule. The whole procedure of this algorithm is described in Algorithm. 3.

We use an example to illustrate the process of rule learning through the Prism algorithm. In particular, the Weather data set[1] is chosen for the illustrative purpose – see Table 1.

As indicated in line 2 of Algorithm 3, a target class needs to selected for each rule to be learned. The Weather data set involves only two classes: 'Yes'

Table 1: Weather Dataset

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

[1] `http://storm.cis.fordham.edu/~gweiss/data-mining/weka-data/weather.nominal.arff`

and 'No', so the Prism algorithm needs to learn a set of rules for each of the two classes.

According to Table 1, we can get a frequency table for each attribute, i.e. we have four frequency tables for the four attributes: Outlook (Table 2), Temperature (Table 3), Humidity (Table 4) and Windy (Table 5).

Table 2: Frequency Table for Outlook

Class label	Outlook= sunny	Outlook= overcast	Outlook= rain
Yes	2	4	3
No	3	0	2
Total	5	4	5

Table 3: Frequency Table for Temperature

Class label	Temperature= hot	Temperature= mild	Temperature= cool
Yes	2	4	3
No	2	2	1
Total	4	6	4

Table 4: Frequency Table for Humidity

Class label	Humidity= high	Humidity= normal
Yes	3	6
No	4	1
Total	7	7

Table 5: Frequency Table for Windy

Class label	Windy= true	Windy= false
Yes	3	6
No	3	2
Total	6	8

Based on the frequency tables, the conditional probabilities for each attribute-

value pair of each attribute can be calculated. We display these here for ease of explanation – in the normal course of the algorithm the probabilities would be calculated when needed, not in advance.

According to Table 2, we can derive the conditional probability for each of the three values of attribute 'Outlook', towards each of the two classes.

$P(Class = Yes|Outlook = sunny) = \frac{2}{5}$
$P(Class = Yes|Outlook = overcast) = \frac{4}{4} = 1$
$P(Class = Yes|Outlook = rain) = \frac{3}{5}$
$P(Class = No|Outlook = sunny) = \frac{3}{5}$
$P(Class = No|Outlook = overcast) = \frac{0}{4} = 0$
$P(Class = No|Outlook = rain) = \frac{2}{5}$

According to Table 3, we can derive the conditional probability for each of the three values of attribute 'Temperature', towards each of the two classes.

$P(Class = Yes|Temperature = hot) = \frac{1}{2}$
$P(Class = Yes|Temperature = mild) = \frac{2}{3}$
$P(Class = Yes|Temperature = cool) = \frac{3}{4}$
$P(Class = No|Temperature = hot) = \frac{1}{2}$
$P(Class = No|Temperature = mild) = \frac{1}{3}$
$P(Class = No|Temperature = cool) = \frac{1}{4}$

According to Table 4, we can derive the conditional probability for each of the two values of attribute 'Humidity', towards each of the two classes.

$P(Class = Yes|Humidity = high) = \frac{3}{7}$
$P(Class = Yes|Humidity = normal) = \frac{6}{7}$
$P(Class = No|Humidity = high) = \frac{4}{7}$
$P(Class = No|Humidity = normal) = \frac{1}{7}$

According to Table 5, we can derive the conditional probability for each of the two values of attribute 'Windy', towards each of the two classes.

$P(Class = Yes|Windy = true) = \frac{1}{2}$
$P(Class = Yes|Windy = false) = \frac{3}{4}$

$P(Class = No|Windy = true) = \frac{1}{2}$
$P(Class = No|Windy = false) = \frac{1}{4}$

When the target class is 'Yes', the first attribute, i.e. Outlook, is selected (line 6 in Algorithm 3) and the attribute-value pair ($Outlook = sunny$ or $Outlook = overcast$ or $Outlook = rain$) with the maximum conditional probability is chosen (line 13 in Algorithm 3). Of the three attribute-value pairs, $Outlook = overcast$ has the maximum conditional probability, i.e. $P(Class = Yes|Outlook = overcast) = 1$. Since the maximum probability is reached, i.e. 1, the learning of the first rule is complete and the first rule learned is expressed as: if $Outlook = overcast$ then $class = yes$. Following the completion of learning the first rule, all four instances with the attribute-value pair $Outlook = overcast$ are deleted from the training set, and the learning of the second rule is started on the reduced training set.

If the first target class is 'No', the attribute-value pair with the maximum conditional probability is $P(Class = No|Outlook = sunny) = \frac{3}{5}$, so the attribute-value pair 'Outlook= sunny' is appended to the left hand side of the first rule being learned. Since the $P(Class = No|Outlook = sunny) = \frac{3}{5}$, the rule confidence has not been 1 yet and thus the learning of the first rule needs to continue. In this context, we need to filter all those training instances that do not meet the condition 'Outlook= sunny' and the reduced the training set is shown in Table 6.

Table 6: Weather data subset comprising 'Outlook= sunny'

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
sunny	mild	normal	true	Yes

On the basis of the reduced training set shown in Table 6, we need to repeat

Table 7: Second Frequency values for Temperature, Humidity and Windy

	Temperature			Humidity		Windy	
Class label	hot	mild	cool	high	normal	true	false
Yes	0	1	1	0	2	1	1
No	2	1	0	3	0	1	2
Total	2	2	1	3	2	2	3

the same procedure for getting a frequency table for each attribute. Since the attribute-value pair 'Outlook= sunny' has been selected and appended to the left hand side of the first rule being learned, the selection of attribute-value pairs is only applied to the other three attributes (Temperature, Humidity and Windy) until the completion of learning the first rule. In other words, the same attribute can not be selected twice towards specializing a rule by appending an attribute-value pair to the left hand side of the rule.

According to Table 6, we can get the frequency for the three attributes as shown in Table 7.

According to Table 7, we can derive the conditional probability for each of the three values of attribute 'Temperature', towards each of the two classes.

$P(Class = Yes|Temperature = hot) = \frac{0}{2} = 0$
$P(Class = Yes|Temperature = mild) = \frac{1}{2}$
$P(Class = Yes|Temperature = cool) = \frac{1}{1}$
$P(Class = No|Temperature = hot) = \frac{2}{2} = 1$
$P(Class = No|Temperature = mild) = \frac{1}{2}$
$P(Class = No|Temperature = cool) = \frac{0}{1}$

According to Table 7, we can derive the conditional probability for each of the two values of attribute 'Humidity', towards each of the two classes.

$P(Class = Yes|Humidity = high) = \frac{0}{3} = 0$

$P(Class = Yes|Humidity = normal) = \frac{2}{2} = 1$
$P(Class = No|Humidity = high) = \frac{3}{3} = 1$
$P(Class = No|Humidity = normal) = \frac{0}{2} = 0$

According to Table 7, we can derive the conditional probability for each of the two values of attribute 'Windy', towards each of the two classes.

$P(Class = Yes|Windy = true) = \frac{1}{2}$
$P(Class = Yes|Windy = false) = \frac{1}{3}$
$P(Class = No|Windy = true) = \frac{1}{2}$
$P(Class = No|Windy = false) = \frac{2}{3}$

According to the above derivation regarding the conditional probability for each of the attribute-value pairs, the ones with the maximum includes $Temperature = hot$, $Temperature = cool$, $Humidity = high$ and $Humidity = normal$, since the conditional probability for each of the attribute-value pairs is 1. However, only one of the attribute-value pairs can be selected at one iteration and the attribute-value pair 'Humidity= high' has the highest frequency (3), so 'Humidity= high' is appended to the left hand side of the current rule being learned. The reason why the attribute-value pair with the highest frequency is selected is due to the case that the support of the rule being learned can be increased while more instances are covered by this rule [38]. So far, the learning of the first rule (with the target class 'No') is complete and the rule is expressed as: if $Outlook = sunny$ and $Humidity = high$ then $class = No$. All the three instances which comprise both $Outlook = sunny$ and $Humidity = high$, are deleted from the training set, such that the learning of the second rule (with the target class 'No') is done on the basis of the reduced training set.

2.4. Heuristic Functions

The heuristics, which are popularly used in decision tree learning and rule learning, are presented in this section for explaining in more detail the essence

of decision tree learning and rule learning.

2.4.1. Entropy

Entropy is a measure of uncertainty, which is typically denoted as S is defined in Eq. 1:

$$S = -\sum_{i=0}^{n} p_i \log_2 p_i \tag{1}$$

where p is the probability that an event occurs and i is the index of the event.

In the context of classification, entropy represented in Eq. 1 can be used to show the initial uncertainty regarding the probability distribution among different classes. For example, in the context of binary classification, if the frequency distribution between two classes is 50/50, then the entropy is maximal, which means the uncertainty has been maximized. Also, if the distribution between two classes is 0/100, then the entropy is 0, which means that there is no uncertainty. We refer to it as initial entropy IS, which is defined in Eq. 2

$$IS = -\sum_{i=0}^{c} p(class_i) \log_2 p(class_i) \tag{2}$$

where $class_i$ represents a class label and i is the index of the class label.

However, in the process of learning a classifier, the measure of uncertainty needs to be done by checking the conditional entropy after given a particular attribute-value pair or checking the average entropy after given a particular attribute. The conditional entropy CS of classifying an instance is defined in Eq. 3:

$$CS(A_x = v_j) = -\sum_{i=0}^{c} p(class_i|A_x = v_j) \log_2 p(class_i|A_x = v_j) \tag{3}$$

where A_x represents an attribute; x is the index of the attribute, v_j is a value of the attribute A_x and j is the index of the attribute value. Also, $p(class_i|A_x = v_j)$ is read as the conditional probability of classifying an instance to $class_i$ given that $A_x = v_j$.

Based on the conditional probability defined in Eq. 3, the average entropy AS of classifying an instance given attribute A_x is defined in Eq.4:

$$AS(A_x) = \sum_{j=0}^{n} w(A_x = v_j) CS(A_x = v_j) \tag{4}$$

where $w(A_x = v_j)$ is the probability that the value of attribute A_x is v_j.

2.4.2. Information Gain

Information gain measures the amount of information gained from training data after an attribute is selected leading to the partition of the training data. Based on the average entropy defined in Eq.4, we can calculate the information gain as defined in Eq. 5:

$$Gain(A_x) = IS - AS(A_x) \tag{5}$$

According to Eq. 5, information gain reflects the amount of the reduction of entropy following the partition of a training set on an attribute, i.e. the higher information gain the larger the amount of reduction of uncertainty.

2.4.3. Gini-Index

Gini-index is a measure of purity, which is defined in Eq. 6:

$$Gini = 1 - \sum_{i=0}^{n} p_i^2 \tag{6}$$

In the context of classification, if a data set contains all instances that belong to the same class, then the data set would be considered of maximum purity. We refer to the measure of purity regarding a data set as initial Gini-index IG, which is defined in Eq. 7:

$$IG = 1 - \sum_{i=0}^{c} p(class_i)^2 \tag{7}$$

However, in the process of learning a classifier, the measure of purity needs to be done by checking the conditional Gini-index after given a particular attribute-

value pair or the average Gini-index after given a particular attribute. The conditional Gini-index CG is defined in Eq. 8:

$$CG(A_x = v_j) = 1 - \sum_{i=0}^{c} p(class_i|A_x = v_j)^2 \tag{8}$$

Based on the defined in Eq. 8, the average Gini-index AG is defined in Eq. 9:

$$AG(A_x) = \sum_{j=0}^{n} w(A_x = v_j)CG(A_x = v_j) \tag{9}$$

In the CART algorithm, the average Gini-index AG is calculated by turning a multi-valued attribute a binary attribute. In this context, Eq. 9 needs to be modified to Eq. 10:

$$AG'(A_x) = w(A_x = v_j)CG(A_x = v_j) + w(A_x \neq v_j)CG(A_x \neq v_j) \tag{10}$$

As mentioned above, each multi-valued attribute needs to be transformed into a binary attribute. Since a multi-valued attribute has n values, the binarization of the attribute involves n ways. In this context, the way of binarising a multi-valued attribute that leads to the minimum average Gini-index will be used to partition the training set in the CART algorithm. For example, the Temperature attribute of the Weather data set has three values: 'hot', 'mild' and 'cool'. Therefore, the attribute can be binarised in three ways. The first way is to have the two values 'hot' and 'not hot', while the value 'not hot' is the combination of the other two values 'mild' and 'cool'. Similarly, the second way is to have the values 'mild' and 'not mild', and the third way is to have the values 'cool' and 'not cool'.

According to Eq. 8, the conditional Gini-Index for the above six values is calculated as follows:

$CG(Temperature = hot) = 1 - [(\frac{2}{4})^2 + (\frac{2}{4})^2] = \frac{1}{2}$
$CG(Temperature = not\_hot) = 1 - [(\frac{7}{10})^2 + (\frac{3}{10})^2] = \frac{29}{50}$
$CG(Temperature = mild) = 1 - [(\frac{3}{4})^2 + (\frac{1}{4})^2] = \frac{3}{8}$

$CG(Temperature = not\_mild) = 1 - [(\frac{5}{8})^2 + (\frac{3}{8})^2] = \frac{17}{32}$
$CG(Temperature = cool) = 1 - [(\frac{3}{4})^2 + (\frac{1}{4})^2] = \frac{3}{8}$
$CG(Temperature = not\_cool) = 1 - [(\frac{3}{5})^2 + (\frac{2}{5})^2] = \frac{13}{25}$

According to Eq. 10, the average Gini-Index for the three ways of attribute binarisation is calculated as follows:

$AG'(Hot) = \frac{2}{7} \times CG(Temperature = hot) + \frac{5}{7} \times CG(Temperature = not\_hot) = \frac{39}{70} = 0.56$
$AG'(Mild) = \frac{3}{7} \times CG(Temperature = mild) + \frac{4}{7} \times CG(Temperature = not\_mild) = \frac{13}{28} = 0.46$
$AG'(Cool) = \frac{2}{7} \times CG(Temperature = cool) + \frac{5}{7} \times CG(Temperature = not\_cool) = \frac{7}{65} + \frac{3}{8} = 0.48$

$AG'(Mild)$ is the minimum over the three ways of attribute binarisation, which indicates that the second way of binarising the temperature attribute (by having the two values 'mild' and 'not mild') is used to partition the training set into two subsets that comprise $Temperature = mild$ and $Temperature = notmild$, respectively. More details on attribute selection in the CART algorithm can be found in [11].

3. Related Work

Decision tree learning has become a powerful approach of machine learning due to the fact that the models learned with this approach are represented in a white box manner. In other words, decision trees are transparent so people can clearly identify how a decision is reached through mapping inputs to outputs [22, 23]. In practice, decision tree learning has been involved in broad areas of application, such as text classification [12], biomedicine [39], intelligent tutoring systems [5] and transient stability assessment [36].

In terms of algorithmic development, decision tree learning has become com-

petitive since Quinlan developed the ID3 algorithm [32] with very good performance especially on the chess end games data set [31]. However, the ID3 algorithm is unable to handle continuous attributes directly so Quinlan developed the C4.5 algorithm as an extension of ID3 for effectively dealing with continuous attributes and replacing missing values [34, 35]. Further to C4.5, Quinlan developed a commercial version of decision tree learning algorithm referred to as C5.0 [14].

Decision tree learning methods were extended through the use of pruning methods and a comparison of different pruning methods was made in [8]. In addition, decision tree learning methods have also been used in the context of ensemble learning [28] for increasing the overall accuracy of classification. Two popular approaches of ensemble learning include Bagging [2] and Boosting [9]. Over the past decade, decision tree learning methods have also been extended in other different ways. One way is by adding cost functions into heuristics for attribute selection towards minimizing the risk of incorrect classification. Some more recent work can be found in [40, 30, 18]. Another popular way of dealing with continuous attributes is through fuzzification towards the generation of fuzzy decision trees. Some more recent work can be found in [17, 1, 16].

The nature of decision tree learning is to generate a set of non-overlapping rules, which constrains that these rules must have at least one common attribute in order to make the rules fit in a tree structure. Due to the above constraint, decision tree learning often results in complex trees being generated and difficulty for people to understand the information extracted from the trees [10]. In order to simplify decision trees, Quinlan investigated in [33] the use of pruning methods, such as reduced error pruning [7]. However, even if decision trees are simplified by using pruning methods, it is still difficult to avoid the case that decision trees are too cumbersome, complex and inscrutable to provide insight into a domain for people to use as knowledge [34, 10]. Also, complex decision trees are more likely to overfit training data than simple trees [10]. On the other hand, Cendrowska pointed out in [4] that the nature of decision tree learning may result in the replicated subtree problem due to the constraint that rules

must have at least one common attribute in order to represent these rules in a tree structure, i.e. rules that have no common attribute can not fit in a tree structure. It is also mentioned in [4] that the replicated subtree problem may result in the need to examine the whole tree for extracting rules about a single classification in the worst case, which makes it difficult to manipulate for expert systems.

In order to overcome the replicated subtree problem, the Prism algorithm has been developed in [4], which has led to the motivation for developing rule learning methods through the separate and conquer strategy. A comprehensive review of rule learning methods can be found in [10]. As shown in Algorithm 3, the nature of the Prism algorithm is to select a target class, and then to learn a rule by selecting attribute-value pairs iteratively for specialising this rule, until all the instances covered by this rule belong to the target class. However, it is generally unknown whether the selected target class can lead to the generation of a high quality rule. In fact, the separate and conquer strategy involves the learning of rules on a sequential basis, i.e. the learning of a rule must not start until the learning of the last rule is completed. In this context, the outcome of learning one rule impacts greatly on the outcome of learning the next rule. In the case of the Prism algorithm, the selected target class may not be suitable for learning a high quality rule, and even leads to the generation of an inconsistent rule, which means that the instances covered by this rule belong to different classes [20, 26]. Also, the learning of the subsequent rules would be impacted greatly, due to the unexpected outcome that the last rule learned is inconsistent. The Prism algorithm introduced in [4] is designed to simply keep selecting the same class as the target class towards learning a set of rules, all of which are assigned this class as the rule consequent, and then repeat the same procedure by having another class as the target class for learning a different set of rules. The above description indicates that the Prism algorithm has a bias on the selection of the target class for a rule to be learned, i.e. ineffective selection of the target class would result in a low quality rule being learned.

In the next section, we will address the above issues of decision tree learning

algorithms and the Prism algorithm by proposing a new algorithm based on Gini-index through the separate and conquer strategy. In particular, we will employ the conditional Gini-index defined in Eq. 8 for the selection of attribute-value pairs to overcome the limitations of attribute selection in decision tree learning algorithms. Also, the proposed rule learning algorithm, which is based on Gini-index, is aimed to overcome the limitations of the Prism algorithm in terms of the bias on the target class selection.

4. Gini-Index Based Rule Generation

In this section, we describe the Gini-Index Based Rule Generation (GIBRG) algorithm and illustrate its procedure on the weather dataset introduces in Section 2. We also argue why Gini-index is more suitable for rule learning than decision tree learning, through analysing the main difference between these two types of approaches in terms of their nature of learning.

4.1. Key Features

The GIBRG algorithm is designed to select an attribute-value pair iteratively based on the conditional Gini-index defined in Eq. 8. In particular, the attribute-value pair, which contributes to maximizing the ability of the rule being learned to discriminate between classes, would be selected. In other words, when the conditional Gini-index given an attribute-value pair is zero, the rule can totally discriminate one class from all the other classes and all the instances covered by this rule belong to the class assigned to the rule. The procedure of the GIBRG algorithm is described in Algorithm 4.

To illustrate the GIBRG algorithm we use the weather dataset displayed in Table 1, and the frequency Tables 2, 3, 4 and 5 from Section 2.

According to Table 2, we can derive the Gini-index for each of the three values of attribute 'Outlook' according to Eq. 8:

$CG(Outlook = sunny) = 1 - [(\frac{2}{5})^2 + (\frac{3}{5})^2] = \frac{12}{25}$
$CG(Outlook = overcast) = 1 - [(\frac{4}{4})^2 + 0] = 0$

Algorithm 4: GIBRG Algorithm

Input : a training set T, a subset $T' \subseteq T$, an attribute set AS, an instance $t \in T$, dimensionality d, an attribute A_x, an attribute value v_{xn}, Gini-index G, class C_i

Output: a rule set RS, a result set of instances T" covered by a rule $R \in RS$

```
1  Initialize: T' = T, T" = T, G = 1;
2  while T' ≠ φ do
3      while G ≠ 0 do
4          x = 0; j = 0; G = 1; while x < d do
5              k = 0;
6              for each value v_xn of A_x do
7                  Calculate G(A_x = v_xn)
8                  if G(A_x = v_xn) < G) then
9                      G = G(A_x = v_xn); j = x; k = n;
10                 end
11             end
12             x++;
13         end
14         assign A_j = v_jk to R as a rule term, when G(A_j = v_jk) is
             minimal; AS= AS - {A_j};
15         d= d − 1; ∀t : T" = T" − {t}, if t ∈ T" and t comprise A_j = v_jk;
16     end
17     RS= RS ∪ {R}; T'= T' - T";
18 end
```

$CG(Outlook = rain) = 1 - [(\frac{3}{5})^2 + (\frac{2}{5})^2] = \frac{12}{25}$

According to Table 3, we can derive the Gini-index for each of the three values of attribute 'Temperature' according to Eq. 8:

$CG(Temperature = hot) = 1 - [(\frac{2}{4})^2 + (\frac{2}{4})^2] = \frac{1}{2}$
$CG(Temperature = mild) = 1 - [(\frac{4}{6})^2 + (\frac{2}{6})^2] = \frac{4}{9}$
$CG(Temperature = cool) = 1 - [(\frac{3}{4})^2 + (\frac{1}{4})^2] = \frac{3}{8}$

According to Table 4, we can derive the Gini-index for each of the two values of attribute 'Humidity' according to Eq. 8:

$CG(Humidity = high) = 1 - [(\frac{3}{7})^2 + (\frac{4}{7})^2] = \frac{24}{49}$
$CG(Humidity = normal) = 1 - [(\frac{6}{7})^2 + (\frac{1}{7})^2] = \frac{12}{49}$

According to Table 5, we can derive the Gini-index for each of the two values of attribute 'Humidity' according to Eq. 8:

$CG(Windy = true) = 1 - [(\frac{3}{6})^2 + (\frac{3}{6})^2] = \frac{1}{2}$
$CG(Windy = false) = 1 - [(\frac{6}{8})^2 + (\frac{2}{8})^2] = \frac{3}{8}$

According to the above derivation regarding the Gini-index for each of the attribute-value pairs, the one with the minimum is $Outlook = overcast$ and the attribute-value pair is appended to the left hand side of the first rule being learned. Since the Gini-index for the attribute-value pair $Outlook = overcast$ is 0, the learning of the first rule is complete. The first rule learned is expressed as: if $Outlook = overcast$ then $class = yes$, as all the four instances, which meet the condition $Outlook = overcast$, are assigned the class 'yes' regarding whether to play or not.

Following the completion of learning the first rule, all the four instances, which comprise the attribute-value pair $Outlook = overcast$, need to be deleted from the training set and then the second rule will be learned on the basis of the rest of the training instances. In particular, after the four instances are deleted from the training set, the reduced training set is shown in Table 8.

In order to learn the second rule, we need to create the following frequency

Table 8: Weather data subset comprising '*Outlook* $\neq$ *overcast*'

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
rain	mild	high	true	No

tables for all of the four attributes:

Table 9: Frequency Table for Outlook in iteration 2 by GIBRG

Class label	Outlook= sunny	Outlook= overcast	Outlook= rain
Yes	2	0	3
No	3	0	2
Total	5	0	5

Table 10: Frequency Table for Temperature in iteration 2 by GIBRG

Class label	Temperature= hot	Temperature= mild	Temperature= cool
Yes	0	3	2
No	2	2	1
Total	2	5	3

Table 11: Frequency Table for Humidity in iteration 2 by GIBRG

Class label	Humidity= high	Humidity= normal
Yes	1	4
No	4	1
Total	5	5

Table 12: Frequency Table for Windy in iteration 2 by GIBRG

Class label	Windy= true	Windy= false
Yes	1	4
No	3	2
Total	4	6

According to Table 9, we can derive the Gini-index for each of the three values of attribute 'Outlook' according to Eq. 8:

$CG(Outlook = sunny) = 1 - [(\frac{2}{5})^2 + (\frac{3}{5})^2] = \frac{12}{25}$
$CG(Outlook = overcast) = N/A$
$CG(Outlook = rain) = 1 - [(\frac{3}{5})^2 + (\frac{2}{5})^2] = \frac{12}{25}$

According to Table 10, we can derive the Gini-index for each of the three values of attribute 'Temperature' according to Eq. 8:

$CG(Temperature = hot) = 1 - [(\frac{0}{2})^2 + (\frac{2}{2})^2] = 0$
$CG(Temperature = mild) = 1 - [(\frac{3}{5})^2 + (\frac{2}{5})^2] = \frac{12}{25}$
$CG(Temperature = cool) = 1 - [(\frac{2}{3})^2 + (\frac{1}{3})^2] = \frac{4}{9}$

According to Table 11, we can derive the Gini-index for each of the two values of attribute 'Humidity' according to Eq. 8:

$CG(Humidity = high) = 1 - [(\frac{1}{5})^2 + (\frac{4}{5})^2] = \frac{8}{25}$

$CG(Humidity = normal) = 1 - [(\frac{4}{5})^2 + (\frac{1}{5})^2] = \frac{8}{25}$

According to Table 12, we can derive the Gini-index for each of the two values of attribute 'Humidity' according to Eq. 8:

$CG(Windy = true) = 1 - [(\frac{1}{4})^2 + (\frac{3}{4})^2] = \frac{3}{8}$
$CG(Windy = false) = 1 - [(\frac{4}{6})^2 + (\frac{2}{6})^2] = \frac{4}{9}$

According to the above derivation regarding the Gini-index for each of the attribute-value pairs, the one with the minimum is $Temperature = hot$ and the attribute-value pair is appended to the left hand side of the first rule being learned. Since the Gini-index for the attribute-value pair $Temperature = hot$ is 0, the learning of the second rule is complete. The second rule learned is expressed as: if $Temperature = hot$ then $class = No$.

Following the completion of learning the second rule, the two instances which comprise $Temperature = hot$, need to be deleted from the training set and then the third rule will be learned on the basis of the rest of the training instances, and so on until all the training instances have been covered by at least one of the learned rules.

4.2. Justification

As mentioned in Section 1, the nature of rule learning is at selection of attribute-value pairs towards specialising a rule, whereas the nature of decision tree learning is at attribute selection. In other words, decision tree learning is attribute oriented whereas rule learning is attribute-value oriented. Due to their difference in terms of the nature of learning, rule learning is considered to be more suitable than decision tree learning in some specific cases. In particular, decision tree learning involves evaluation of the ability of each attribute to determine the value of the class attribute. However, as argued in [4], an attribute may be highly relevant to only one class but irrelevant to all the other classes. Also, it is highly possible that only one value of an attribute is relevant

to classifying instances to a particular class. The above two points have led to the motivation of developing the Prism algorithm and the recommendation of taking advantage of the separate and conquer strategy for rule learning.

As argued in Section 1, decision tree learning involves measuring the importance of an attribute towards increasing the overall quality of a sub-tree being learned. As mentioned in Section 1, a decision tree can be converted into a set of rules. In this context, decision tree learning can only lead to the production of a set of rules of high quality on average, without the guarantee that each single rule is of high quality. In other words, the case that a set of rules is of high quality on average does not mean that each single rule is of good quality. In contrast, rule learning involves measuring the importance of an attribute-value pair towards increasing the quality of each single rule being learned. For example, as introduced in Section 2, Gini-index can be used to measure the importance of an attribute-value pair towards increasing the ability of a rule to discriminate between classes, i.e. the degree to which a rule is biased towards one class and against the other classes. In the CART algorithm, the use of average Gini-index defined in Eq. 10 is to measure the importance of an attribute towards increasing the overall ability of a sub-tree being learned to discriminate between classes. However, as mentioned in Section 1, it is highly possible that only one value of an attribute is of high importance towards classifying instances. In this case, the learned decision tree may provide some rules (extracted from some tree branches) highly capable of discriminating between classes, due to the inclusion of the attribute-value pairs of high importance, but the tree may provide other rules poorly capable of discriminating between classes, due to inclusion of the attribute-value pairs of low importance. In real applications, it is critical in rule based classification that each single rule is highly discriminative, since an instance is typically classified by using a single rule. From this point of view, it is more suitable to employ Gini-index in rule learning, which leads to the motivation of proposing the GIBRG algorithm.

On the other hand, as mentioned in Section 1, the nature of decision tree learning could result in the replicated sub-tree problem, which means that a set

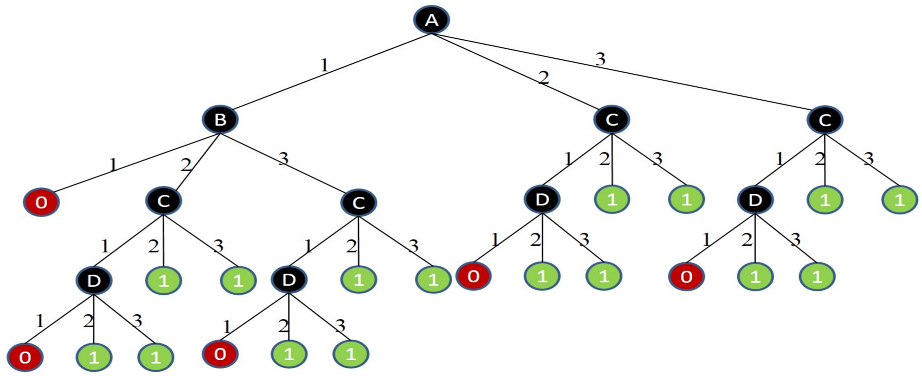

Figure 1: Cendrowska's Replicated Subtree Problem [19]

of rules extracted from a decision tree contains a lot of redundant rule terms. For example, as illustrated in Fig. 1, the decision tree contains 21 rules and 62 terms. However, if all the redundancy is removed from the decision tree according to [4], the extracted rule set would just contain 2 rules and 2 terms as follows:

- Rule 1: if $a = 1$ and $b = 1$ then $class = 0$;
- Rule 2: if $c = 1$ and $d = 1$ then $class = 0$;

Also, as reported in [24], through learning from the Contact Lenses Data set [4], ID3 produces 3 rules and 5 rule terms as follows:

- Rule 1: if $tearproductionrate = reduced$ then $class = nolenses$;
- Rule 2: if $tearproductionrate = normal$ and $Astigmatic = yes$ then $class = Hardlenses$;
- Rule 3: if $tearproductionrate = normal$ and $Astigmatic = no$ then $class = Softlenses$;

However, IEBRG produces 3 rules and 3 rule terms as follows:

- Rule 1: if $tearproductionrate = reduced$ then $class = nolenses$;

- Rule 2: if $Astigmatic = yes$ then $class = Hardlenses$;
- Rule 3: if $Astigmatic = no$ then $class = Softlenses$;

Through comparing the two rule sets produced by the two algorithms respectively, it can be indicated explicitly that the nature of the ID3 algorithm leads to a redundant term ($tearproductionrate = normal$) being produced, which results in the production of a more complex model, in comparison with IEBRG. In fact, the *tearproductionrate* attribute has only two values, so the value of this attribute must be 'normal', if the first rule ($iftearproductionrate = reducedthenclass = nolenses$) does not fire. In this case, there is no need to examine the value of this attribute in the following rules (Rule 2 and Rule 3). The above argumentation could indicate that the divide and conquer strategy would usually lead to the production of a higher number of more complex rules than the separate and conquer strategy. We will show experimental results in Section 5.2 in terms of model complexity to support this argumentation.

As mentioned in Section 3, the Prism algorithm has a bias on the selection of a target class for learning a rule that has the selected target class as the consequent of this rule. The bias can lead to the generation of a low quality rule, which also impacts negatively on the learning of all the subsequent rules. In contrast, the GIBRG algorithm does not need to have a target class selected for learning a rule, but it can generally lead to the outcome that the left hand side of the rule being learned is eventually mapped to a single class, i.e. it is an eventual outcome that all the training instances covered by the rule being learned belong to one class only and the class is assigned to the right hand side of the rule as the consequent.

In addition, as illustrated in Section 4.1, the Prism algorithm is aimed at selecting each class in turn as the target class, towards learning a set of rules of this class from the whole training set. This way of rule learning can lead to an increase in the number of rules and terms, since the learning task could be considered as the case that n (i.e. the number of classes) rule based classifiers are learned from the same training set, in comparison with the GIBRG algorithm.

Moreover, the nature of the Prism algorithm could also lead to the production of more complex rule based classifiers than decision tree learning algorithms, especially when the training set is larger and the number of classes is higher. The above argumentation will be validated empirically through experimental studies.

5. Experimental Study, Results and Discussion

In this section, we report an experimental study to validate the GIBRG algorithm by comparing it with C4.5, CART and Prism in terms of classification accuracy and number of rules and terms.

We choose to compare with the C4.5 algorithm as it is the most popular one of the decision tree learning algorithms. The CART algorithm employs Gini-index for attribute selection, so we choose to compare it with the GIBRG algorithm in order to show that it is more suitable for Gini-index to be used in rule learning than in decision tree learning. The Prism algorithm is the representative of rule learning algorithms so we compare it with the GIBRG algorithm in order to show the competitivity of the latter algorithm in the context of separate and conquer rule learning.

In terms of classification accuracy, the experiments are conducted by partitioning a data set into a training set and a test set in the ratio of 70:30. On each data set, the experiment is repeated 100 times (in terms of data partitioning) and the average accuracy is taken for comparative validation. In terms of number of rules and terms, each whole data set is used for learning a rule based classifier towards counting rules and terms. The above procedures are followed for 20 data sets from the UCI repository, which are used for testing the performance of the chosen algorithms. The characteristics of the 20 data sets are described in Table 13.

Furthermore, we report an extended study on rule pruning to show how effectively overfitting can be avoided in the case of noisy data. In particular, we conducted experiments using 3 out of the 20 UCI data sets shown in Ta-

Table 13: Data sets

Dataset	Attribute Types	#Attributes	#Instances	#Classes
anneal	discrete, continuous	38	798	6
balance-scale	discrete	4	625	3
breast-cancer	discrete	9	286	2
breast-w	continuous	10	699	2
credit-a	discrete, continuous	15	690	2
credit-g	discrete, continuous	20	1000	2
cylinder-bands	discrete, continuous	40	540	2
dermatology	discrete, continuous	35	366	6
diabetes	discrete, continuous	20	768	2
hepatitis	discrete, continuous	20	155	2
ionosphere	continuous	34	351	2
iris	continuous	4	150	3
kr-vs-kp	discrete	36	3196	2
labor	discrete, continuous	17	57	2
lymph	discrete, continuous	19	148	4
mushroom	discrete	22	8124	2
tae	discrete, continuous	6	151	3
vote	discrete	16	435	2
wine	continuous	13	178	3
zoo	discrete, continuous	18	101	7

ble 13, i.e. 'breast-cancer', 'hepatitis' and 'vote', as well as the 'heart-stalog' and 'car' data sets. The heart-stalog data set contains 13 (discrete or continuous) attributes, 270 instances and 3 classes, whereas the car data set contains 6 discrete attributes, 1728 instances and 4 classes. The experiments are conducted in the same way as above, i.e. 70:30 data partitioning with 100 repetitions on each data set.

5.1. Accuracy comparison

The experimental results are presented in Table 14 in terms of the classification accuracy performed by the chosen algorithms mentioned above. The accuracy results are also displayed in Figure 2 for easy visual comparison. The GIBRG algorithm outperforms the other three algorithms, namely, C4.5, CART and Prism, in 12 out of 20 cases. Also, there are three other cases (i.e. on the three data sets 'kr-vs-kp', 'mushroom' and 'vote') that GIBRG performs the same as C4.5 or CART. Regarding the performance on the two data sets 'balance-scale' and 'dermatology', the GIBRG algorithm just shows to be marginally worse than C4.5, CART or Prism.

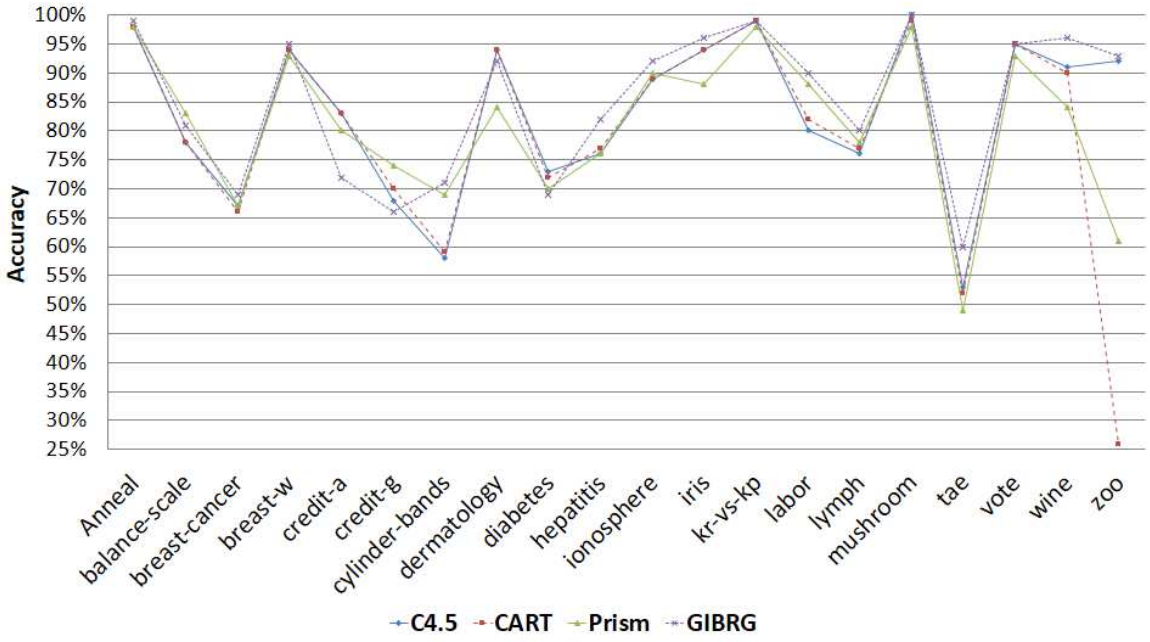

Figure 2: Accuracy comparison

To further investigate the performance of the GIBRG algorithm, we apply the Wilcoxon sign rank test to investigate if the average performance of GIBRG classification accuracy is higher than the average classification accuracy of C4.5, CART and Prism, respectively. The Wilcoxon sign rank test is used, as it

Table 14: Classification Accuracy

Dataset	C4.5	CART	Prism	GIBRG
anneal	98%	98%	98%	99%
balance-scale	78%	79%	83%	81%
breast-cancer	67%	66%	67%	66%
breast-w	94%	94%	93%	95%
credit-a	83%	83%	80%	72%
credit-g	68%	70%	74%	66%
cylinder-bands	58%	59%	69%	71%
dermatology	94%	94%	84%	92%
diabetes	72%	72%	70%	69%
hepatitis	76%	77%	76%	82%
ionosphere	89%	89%	90%	92%
iris	94%	94%	88%	96%
kr-vs-kp	99%	99%	98%	99%
labor	80%	82%	88%	90%
lymph	76%	77%	78%	80%
mushroom	100%	99%	98%	100%
tae	53%	52%	49%	60%
vote	95%	95%	93%	95%
wine	91%	90%	84%	96%
zoo	92%	26%	61%	93%

has been shown to be more appropriate than the use of the paired t-test as documented in [6].

The Wilcoxon sign rank test [6] allows the comparison between 2 classifiers across several datasets. The null hypothesis is that the performance of the two classifiers is not significantly different. The differences in performance between classifiers are calculated and ranked. Two sums of ranks are then calculated – one for the positive differences and one for the negative differences. The

smaller of the sums is compared with the critical value from the table of exact critical values for the Wilcoxons test, for a particular significance level (e.g. $\alpha < 0.05$ or $\alpha < 0.01$) and the number of datasets (N), which in our case is 20. Table 15 displays details of the Wilcoxon's test for each pair of classifiers in our experiments; we used $\alpha < 0.05$ (one-tailed test) and $N = 20$.

Table 15: Wilcoxon sign rank tests

Compared classifiers	Smaller sum	Critical value	Null hypothesis accept/reject
GIBRG - C4.5	49.5	60	Reject
GIBRG - CART	47.5	60	Reject
GIBRG - Prism	38.5	60	Reject

The Wilcoxon rank test results indicate that the null hypothesis should be rejected, meaning that the average GIBRG performance is higher than each of the other three algorithms and that this difference is unlikely to occur by chance. In other words, the GIBRG algorithm performs significantly better than C4.5, CART and Prism algorithms in term of accuracy.

Through comparing the performance of GIBRG with Prism, the results indicate that the bias resulting from the selection of a target class for a rule to be learned by using Prism can result in the generation of a low quality rule. Also, ineffective selection of target classes can also result in the learned rules being inconsistent. Moreover, the above learning outcomes would also impact greatly on the learning of any subsequent rules, as mentioned in Section 4.2. However, the GIBRG algorithm does not have the bias on the selection of target classes and outperforms the Prism algorithm in 16 out of 20 cases shown in Table 14. Therefore, in the context of rule learning through the separate and conquer strategy, it is more suitable to involve the selection of attribute-value pairs for learning a rule without the need to ensure that the rule must be assigned a particular class as the rule consequent.

The results shown in Table 14 and the Wilcoxon rank test support the argument that the Gini-index is more suitable for rule learning than decision tree

learning. As mentioned in Section 4.2, Gini-index can be used to measure the importance of an attribute-value pair towards increasing the discriminability (the ability to discriminate between classes) of a rule being learned. The CART algorithm employs Gini-index for measuring the average importance of the values of an attribute, towards increasing on average the discriminability of rules extracted from different branches of a tree. This way of using Gini-index would lead to the outcome that the learned decision tree provides some rules of higher quality but the others of lower quality. However, as mentioned in Section 1, an unseen instance is typically classified by using a single rule that fires. In this context, if the rule used to classify an instance is of low quality, then it would be very likely to give a wrong classification. Therefore, it is critical to ensure that each single rule learned is of high quality, which indicates that the nature of the GIBRG algorithm has led to a better way of using Gini-index than the nature of the CART algorithm towards increasing the discriminability of each single rule.

On the other hand, through comparing GIBRG with C4.5, the results indicate that the nature of rule learning can lead to a more effective measure than the nature of decision tree learning, towards increasing the quality of each single rule for providing accurate classification of unseen instances. As mentioned in Section 4.2, an attribute may not necessarily have all its values relevant to be used for classifying instances. In other words, it is highly possible that only some (but not all) values of an attribute can result in high discrimination between different classes, especially when an attribute is highly imbalanced by means of some values of this attribute of high frequency but the others of low frequency. In this context, the nature of decision tree learning, which is aimed at the selection of a whole attribute on a recursive basis, would result in a decision tree having irrelevant branches due to the selection of irrelevant attribute values.

5.2. Rules and terms comparison

The above case of having irrelevant branches in a decision tree can also lead

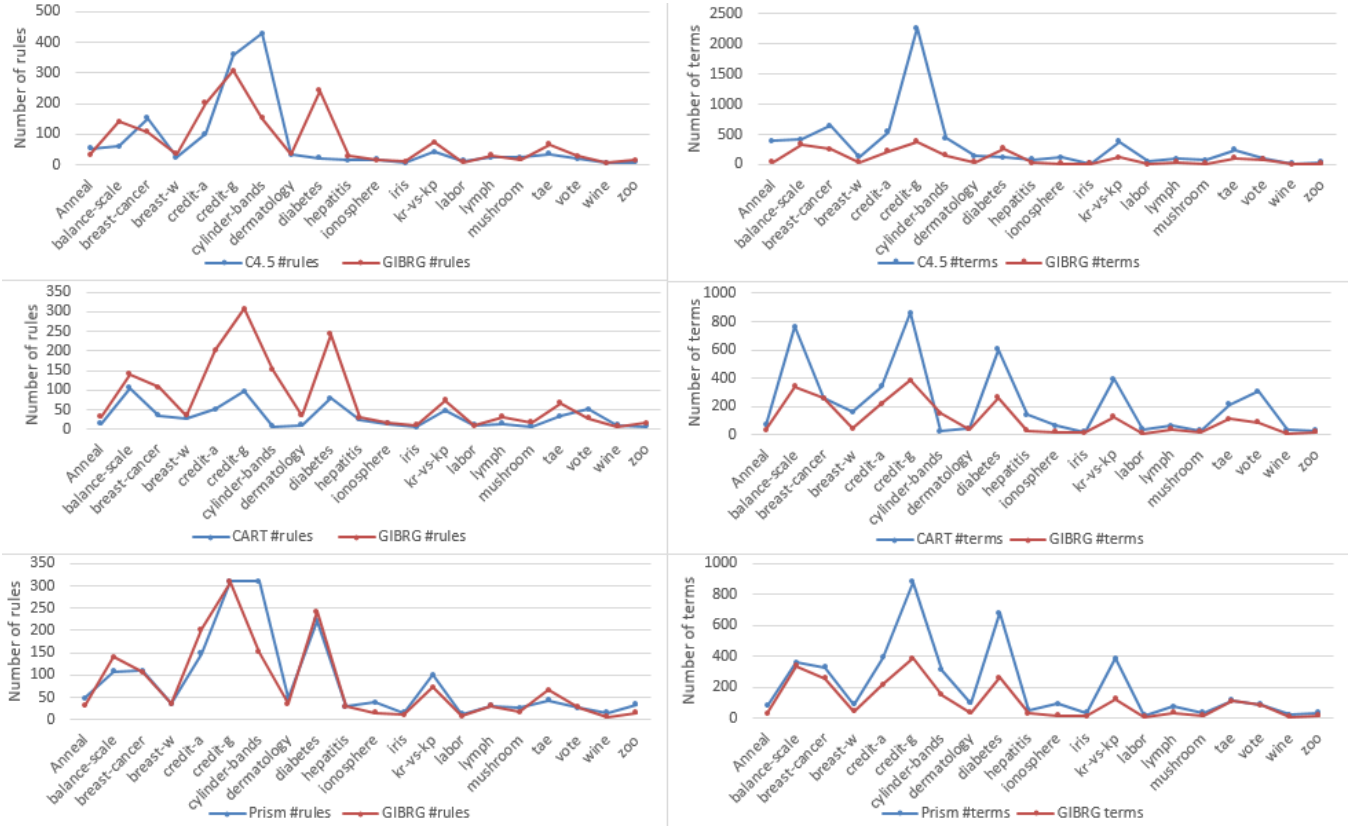

Figure 3: Rules and terms comparison

to the outcome that decision tree learning algorithms produce more complex rule based classifiers than rule learning algorithms. For example, as mentioned in Section 4.2, the contact lenses data set contains only 4 attributes and 24 instances. ID3 (a decision tree learning algorithm) produced 3 rules and 5 terms, whereas IEBRG (a rule learning algorithm) produced 3 rules and 3 terms.

Table 16 displays the number of rules and terms for each of the 4 algorithms on all 20 datasets. Figure 3 display graphs comparing GIBRG with C4.5, CART and Prism, respectively, with regards to the number of rules and terms.

The results shown in Table 16 and Figure 3 indicate that the nature of the GIBRG algorithm leads to the reduction of the complexity of rule based classifiers in comparison with the other algorithms in most cases.

Table 16: Number of rules and terms

Dataset	C4.5		CART		Prism		GIBRG	
	#rules	#terms	#rules	#terms	#rules	#terms	#rules	#terms
anneal	53	390	14	68	48	81	32	32
balance-scale	60	419	105	763	109	359	141	338
breast-cancer	152	645	35	257	110	329	107	256
breast-w	23	124	28	159	36	87	36	42
credit-a	101	546	51	342	148	392	202	217
credit-g	359	2262	96	863	310	882	308	386
cylinder-bands	430	432	7	21	310	312	152	152
dermatology	33	141	10	44	48	95	36	36
diabetes	22	120	80	604	223	680	243	263
hepatitis	16	81	25	142	30	51	29	29
ionosphere	18	121	14	63	39	92	16	16
iris	5	16	6	18	16	35	11	11
kr-vs-kp	43	379	47	392	101	387	73	123
labor	13	44	10	36	12	15	8	8
lymph	23	93	14	61	32	74	31	37
mushroom	25	67	7	26	27	34	17	17
tae	35	242	34	213	44	115	67	109
vote	19	97	51	304	27	88	28	86
wine	5	12	10	35	16	21	7	7
zoo	9	38	7	26	35	35	16	16

In particular, GIBRG is capable of producing simpler rule based classifiers, when compared with the Prism algorithm – the results in Table 16 and Figure 3 indicate that GIBRG produces a smaller number of rules and terms than Prism in 15 out of 20 cases. In fact, the complexity of a rule based classifier is determined directly from the number of rule terms, since it is needed to examine each single rule term towards classifying an unseen instance. From this point of

view, GIBRG produces a simpler rule based classifier (a smaller number of rule terms) than Prism in all of the 20 cases. In comparison with C4.5 and CART, the GIBRG produces simpler rule based classifiers in terms of number of rule terms in 18 out of 20 cases.

In addition, the results also show that GIBRG produce a larger number of rules than CART in most cases. This could be explained partially by the nature of CART algorithm in terms of attribute selection. As mentioned in Section 2.4, the CART algorithm is aimed at production of a binary tree. In particular, each attribute needs to be binarized by having one value of the attribute as the positive value and having all the other values merged as the negative value. The binarization of attributes can greatly impact on the number of rules. As analyzed in [24], the number of rules could be affected by both the number of attributes and attribute complexity. From this point of view, the binarization of attributes can lead to reduction of the number of rules learned from a data set. For the CART algorithm, when an attribute is selected, it has been binarized, i.e. it contains only two values leading to two branches from a node of a decision tree, so the number of rules could be reduced accordingly. However, the number of terms is more influential than the number of rules in terms of the complexity of a classifier, as mentioned above.

5.3. Pruning Comparison

In this study, we compare the four learning algorithms (C4.5, CART, Prism and GIBRG) both with and without pruning, in terms of classification accuracy. For C4.5 and CART, the Reduced Error Pruning (REP) and the Cost Complexity Pruning (CCP) algorithms are adopted, due to the fact that REP and CCP have been popularly used respectively for pruning decision trees produced by C4.5 and CART [33]. For Prism and GIBRG, the Jmid-pruning algorithm is adopted, since this algorithm has been used successfully for pruning rules produced by Prism [21, 27], and GIBRG is a similar rule learning algorithm that follows the separate and conquer strategy.

The three data sets 'breast-cancer', 'heart-stalog' and 'hepatitis' are chosen,

since they are all bio-medical ones and are known to be noisy. The 'vote' data set contains missing values, and we replaced them with the most frequently occurring (majority) values of the corresponding attributes, which could naturally introduce noise into the data set, i.e. the most frequently occurring value is not guaranteed to be correct. The 'Car' data set is not very noisy but was also used to show how pruning algorithms can impact the classification accuracy [21, 29].

The results are shown in Table 17, both with and without pruning for each learning algorithm on each data set. The results indicate that the classification performance of GIBRG is either improved or remains unchanged in all the 5 cases, after the Jmid-pruning algorithm is used.

Table 17: Pruning Results

Dataset	C4.5		CART		Prism		GIBRG	
	unpruned	pruned	unpruned	pruned	unpruned	pruned	unpruned	pruned
car	92%	88%	97%	97%	90%	89%	96%	96%
breast-cancer	67%	71%	66%	70%	67%	72%	66%	69%
heart-stalog	77%	77%	76%	77%	75%	74%	62%	65%
hepatitis	76%	79%	77%	79%	76%	78%	82%	84%
vote	95%	96%	95%	96%	93%	95%	95%	95%

On the 'car' data set, the adoption of pruning leads to decrease in the classification performance of C4.5 and Prism, which generally indicates that pruning could result in underfitting when a data set is not very noisy. In this situation, GIBRG (with the adoption of pruning) still holds without loss of classification accuracy, which could be seen as a really encouraging phenomenon.

On the three bio-medical data sets ('breast-cancer', 'heart-stalog' and 'hepatitis'), the adoption of pruning helps effectively all of the four learning algorithms produce rules of higher quality, leading to advances in classification accuracy, except for Prism with Jmid-pruning on the 'heart-stalog' data set, where the classification accuracy is slightly decreased. Since all the three data sets are really noisy, the above phenomenon indicates that Jmid-pruning can re-

ally help the GIBRG algorithm avoid overfitting and improve the classification performance.

On the 'vote' data set, the performance of GIBRG remains unchanged after the adoption of pruning, whereas the performance of the other three algorithms is slightly improved. Since noise is introduced naturally through replacement of missing values with the most frequently occurring values of the corresponding discrete attributes, the above phenomenon could indicate that the rule based classifiers produced by C4.5, CART and Prism overfit a little bit. In contrast, the rule based classifier produced by GIBRG does not seem to overfit and the adoption of pruning does not result in underfitting either, which could be seen as a really encouraging phenomenon.

Furthermore, the results also show that the adoption of pruning does not lead to decrease in the classification performance of CART and GIBRG in all of the 5 cases, which really encourages the use of Gini-index as a heuristic for rule learning. In other words, when Gini-index based learning algorithms (on their own) happen to produce rules that do not overfit, it would be highly expected that the adoption of pruning algorithms does not result in underfitting.

6. Conclusions

In this paper, we proposed a Gini-index based rule learning algorithm, which outperforms the most popular decision tree learning algorithms (C4.5 and CART) and the representative of rule learning algorithms (Prism) in most cases of the experimental study reported in Section 5.

We proved through the experimental studies that Gini-index is more suitable to be used in rule learning than in decision tree learning, due to the difference between the two types of approaches in terms of their nature of learning. In particular, decision tree learning is attribute oriented, which indicates that attribute selection is aimed at recursively maximizing the discriminability of rules on average without the guarantee that the discriminability of each single rule is increased. In contrast, rule learning is attribute-value oriented, which indicates

that the selection of attribute-value pairs is aimed at continuously maximizing the discriminability of each single rule being learned. In fact, it is critical in real applications that each single rule is of as higher discriminability as possible towards classifying instances covered by this rule to a single class. The experimental results indicated that the proposed algorithm (GIBRG) not only led to improvement in classification accuracy but also to the reduction of the complexity of the learned rule based classifiers, in comparison with decision tree learning algorithms.

On the other hand, we identified the limitation of the Prism algorithm, which is the bias on the target class selection. The bias could result in the production of inconsistent rules of low quality, which are likely to provide unseen instances with incorrect classifications. Also, the need to select each class in turn as the target class for learning a rule set could lead to production of more complex rule based classifiers. We argued that the nature of the GIBRG algorithm would lead to the reduction of the bias, and the experimental results indicated that the GIBRG algorithm led to an improvement in terms of classification accuracy, in comparison with the Prism algorithm. In addition, the nature of the GIBRG algorithm also led to the reduction of the complexity of the learned rule based classifiers.

In future, we will investigate the strategies of rule learning in more depth. For example, the selection of attribute-value pairs will be based on multiple heuristics towards increasing the quality of each single rule. We will also investigate how to select the target class for a rule to be learned towards advancing the Prism algorithm. In addition, we will investigate in more depth the use of pruning algorithms towards advancing the performance of GIBRG.

Acknowledgements

The authors acknowledge support for the research reported in this paper through the Research Development Fund at the University of Portsmouth.

References

[1] A. Altay and D. Cinar. Fuzzy decision trees. In C. Kahraman and zgr Kabak, editors, *Fuzzy Statistical Decision-Making: Theory and Applications*, volume 343, pages 221–261, 2016.

[2] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[3] L. Breiman, J. Friedman, C. J. Stone, and R. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, Monterey, CA, 1984.

[4] J. Cendrowska. Prism: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27:349–370, May 1987.

[5] K. Crockett, A. Latham, and N. Whitton. On predicting learning styles in conversational intelligent tutoring systems using fuzzy decision trees. *International Journal of Human-Computer Studies*, 97:98–115, January 2017.

[6] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7:1–30, 2006.

[7] T. Elomaa and M. Kriinen. An analysis of reduced error pruning. *Journal of Artificial Intelligence Research*, 15(1):163–187, July 2001.

[8] F. Esposito, D. Malerba, and G. Semeraro. Simplifying decision trees by pruning and grafting: New results. In *8th European Conference on Machine Learning*, volume 912, pages 287–290, Crete, Greece, 25-27 April 1995.

[9] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *13th International Conference on Machine Learning*, pages 148–156, Bari, Italy, 3-6 July 1996.

[10] J. Furnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13:3–54, 1999.

[11] J. Han and M. Kamber. *Data Mining: Concepts and Techniques, 2nd ed.* Morgan Kaufmann Publishers, San Francisco, 2006.

[12] K. Khan, R. U. khan, A. Alkhalifah, and N. Ahmad. Urdu text classification using decision trees. In *International Conference on High-Capacity Optical Networks and Enabling/Emerging Technologies*, pages 56–59, Islamabad, Pakistan, 21-23 December 2015.

[13] I. Kononenko and M. Kukar. *Machine Learning and Data Mining: Introduction to Principles and Algorithms*. Horwood Publishing Limited, Chichester, West Sussex, 2007.

[14] M. Kuhn and K. Johnson. *Applied Predictive Modeling*. Springer, New York, 2013.

[15] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[16] Y. Lertworaprachaya, Y. Yang, and R. John. Interval-valued fuzzy decision trees. In *IEEE International Conference on Fuzzy Systems*, pages 1–7, Barcelona, Spain, 18-23 July 2010.

[17] Y. Lertworaprachaya, Y. Yang, and R. John. Interval-valued fuzzy decision trees with optimal neighbourhood perimeter. *Applied Soft Computing*, 24: 851–866, November 2014.

[18] X. Li, H. Zhao, and W. Zhu. A cost sensitive decision tree algorithm with two adaptive mechanisms. *Knowledge-Based Systems*, 88:24–33, November 2015.

[19] H. Liu and M. Cocea. Fuzzy information granulation towards interpretable sentiment analysis. *Granular Computing*, 3(1), 2018.

[20] H. Liu and A. Gegov. Induction of modular classification rules by information entropy based rule generation. In V. Sgurev, R. R. Yager, J. Kacprzyk, and V. Jotsov, editors, *Innovative Issues in Intelligent Systems*, volume 623, pages 217–230, 2016.

[21] H. Liu, A. Gegov, and F. Stahl. J-measure based hybrid pruning for complexity reduction in classification rules. *WSEAS Transactions on Systems*, 12(9):433–446, 2013.

[22] H. Liu, A. Gegov, and M. Cocea. Network based rule representation for knowledge discovery and predictive modelling. In *IEEE International Conference on Fuzzy Systems*, pages 1–8, Istanbul, Turkey, 2-5 August 2015.

[23] H. Liu, M. Cocea, and A. Gegov. Interpretability of computational models for sentiment analysis. In W. Pedrycz and S.-M. Chen, editors, *Sentiment Analysis and Ontology Engineering: An Environment of Computational Intelligence*, volume 639, pages 199–220, 2016.

[24] H. Liu, A. Gegov, and M. Cocea. Complexity control in rule based models for classification in machine learning context. In *UK Workshop on Computational Intelligence*, pages 125–143, Lancaster, UK, 7-9 September 2016.

[25] H. Liu, A. Gegov, and M. Cocea. *Rule Based Systems for Big Data: A Machine Learning Approach*. Springer, Switzerland, 2016.

[26] H. Liu, A. Gegov, and M. Cocea. Generation of classification rules. In *Rule Based Systems for Big Data: A Machine Learning Approach*, volume 13, pages 29–42, 2016.

[27] H. Liu, A. Gegov, and M. Cocea. Simplification of classification rules. In *Rule Based Systems for Big Data: A Machine Learning Approach*, volume 13, pages 43–50, 2016.

[28] H. Liu, A. Gegov, and M. Cocea. Ensemble learning approaches. In *Rule Based Systems for Big Data: A Machine Learning Approach*, volume 13, pages 63–73, 2016.

[29] H. Liu, A. Gegov, and M. Cocea. Case studies. In *Rule Based Systems for Big Data: A Machine Learning Approach*, volume 13, pages 81–95, 2016.

[30] F. Min and W. Zhu. A competition strategy to cost-sensitive decision trees. In T. Li, H. S. Nguyen, G. Wang, J. Grzymala-Busse, R. Janicki, A. E. Hassanien, and H. Yu, editors, *Rough Sets and Knowledge Technology: 7th International Conference, RSKT 2012, Chengdu, China, August 17-20, 2012. Proceedings*, volume 7414, pages 359–368, 2012.

[31] J. R. Quinlan. Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 463–482, 1983.

[32] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, March 1986.

[33] J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.

[34] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Francisco, 1993.

[35] J. R. Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, 4:77–90, March 1996.

[36] M. Rahmatian, Y. C. Chen, A. Palizban, A. Moshref, and W. G. Dunford. Transient stability assessment via decision trees and multivariate adaptive regression splines. *Electric Power Systems Research*, 142:320–328, January 2017.

[37] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[38] P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the right objective measure for association analysis. *Information Systems*, 29(4):293–313, 2004.

[39] M. Tayefi, H. Esmaeili, M. S. Karimian, A. A. Zadeh, M. Ebrahimi, M. Safarian, M. Nematy, S. M. R. Parizadeh, G. A. Ferns, and M. Ghayour-

Mobarhan. The application of a decision tree to establish the parameters associated with hypertension. *Computer Methods and Programs in Biomedicine*, 139:83–91, February 2017.

[40] H. Zhao and X. Li. A cost sensitive decision tree algorithm based on weighted class distribution with batch deleting attribute mechanism. *Information Sciences*, 378:303–316, 2017.